

StUF 4.0

Voor de pilot StUF 4.0 heeft Topicus Overheid een Java consumer gerealiseerd. Deze consumer heeft als doel om te valideren of:

1. uit de WSDL's en XSD's eenvoudig code te genereren is,
2. de API eenvoudig is in het gebruik en
3. de objecten eenvoudig op een eigen datamodel te mappen zijn.

De consumer is ontwikkeld als een op zichzelf staande applicatie, niet als onderdeel van een bestaand product van Topicus Overheid. De code kan gebruikt worden tegen het meegeleverde SoapUI-project of tegen de live web service op <http://stuf4.processfive.com/Stuf4SoapService.svc>.

Hieronder wordt ingegaan op de hierboven benoemde doelen. Per doel wordt een vergelijk gemaakt met de huidige werkwijze van StUF 3.10.

Consumer code is geschreven in Java en ondersteunt bevestigingen via SOAP en REST. Hiervoor wordt gebruik gemaakt van Apache CXF. In de consumer zijn alleen de methoden voor het raadplegen van een ingeschreven persoon op basis van BSN geïmplementeerd. Dit is gedaan omdat de online beschikbaar gestelde producer voorgedefinieerde antwoorden teruggeeft, waardoor de resultaten van zoekacties niet gevalideerd kunnen worden.

Codegeneratie

Behoudens verschillen tussen de handmatig gedefinieerde WSDL en de door het .Net framework 'platgeslagen' WSDL is codegeneratie zeer eenvoudig. Topicus Overheid heeft hiervoor gebruik gemaakt van de maven plug-in van CXF. Met standaardconfiguratie van deze plug-in is het mogelijk binnen enkele seconden Java-code te genereren uit de WSDL.

StUF 3.10

In de oude situatie van StUF 3.10 was het niet mogelijk om direct code te genereren uit de WSDL en daaraan gekoppelde XML-Schema's (XSD). Over deze schema's schrijft KING het volgende:

Zo bevat het constructies die het genereren van code bemoeilijken zo niet zelfs onmogelijk maken, met name het voorkomen van restrictions levert veel problemen op. Daarnaast bevatten de schema's componenten die niet gebruikt worden binnen de StUF berichtencatalogus of koppelvlaak op basis waarvan dus bijv. onnodige code gegenereerd wordt. Ook het grote aantal bestanden binnen 1 namespace wil voor sommige software nog wel eens voor problemen zorgen. Met name import en include constructies leveren dan problemen op.

Deze quote komt uit de documentatie die wordt geleverd bij de XSD-resolver (<http://www.gemmaonline.nl/index.php/XSD-Resolver>). De XSD-resolver is een tool

waarmee de problemen met code-generatie opgelost kunnen worden door via een verzameling XSLT stylesheets de XML-Schema's 'plat te slaan' tot één groot schema. Nadat je dit hebt gedaan dien je vervolgens de WSDL aan te passen met een verwijzing naar het nieuwe, platgeslagen, schema. Pas dan kun je code genereren met bijvoorbeeld de maven plug-in die Topicus Overheid gebruikt.

Iemand zonder ervaring met StUF 3.10 en codegeneratie zal hier al snel een dag (of meer) over doen:

- vol goede moed ga je met je codegenerator aan de slag met de WSDL en XSD's
- code genereren lukt niet, je gaat van alles proberen om de problemen op te lossen
- na verwoede pogingen ga je Googelen en dan kom je achter het bestaan van de XSD-resolver
- je gaat uitvogelen hoe de XSD-resolver werkt
- als je doorhebt hoe je hem moet instellen genereer je je eerste platgeslagen XSD
- de WSDL pas je aan met een verwijzing naar de platgeslagen XSD
- eindelijk kun je je code genereren

Eenvoud van de API

De API bestaat uit een aantal functies met ieder één of meer eenvoudige argumenten: String, Boolean, Datum, etc. Het opvragen van een ingeschreven persoon gaat eenvoudig door het aanroepen van de functie 'raadpleegIngeschrevenPersoon' met als argumenten het BSN, een peildatum en een drietal Booleans om aan te geven of je geïnteresseerd bent in historische gegevens omtrent partner, verblijfplaats of verblijfstitel. Dit geldt voor de SOAP-service. In het geval van REST is het nog eenvoudiger, je vraagt de resource 'IngeschrevenPersoon/RaadpleegIngeschrevenPersoon' op met als parameter het BSN. Deze aanroepen zijn eenvoudig te begrijpen en daardoor zijn aanroepen eenvoudig te implementeren.

StUF 3.10

In StUF 3.10 gaan bevestigingen beduidend anders. StUF 3.10 is ontworpen als een volledig naar eigen wensen aan te passen querytaal. Als gevolg hiervan moet je om een ingeschreven persoon te raadplegen een vraag stellen waarin je tot op veldniveau exact dient te beschrijven wat je in het antwoord terug wil zien. Daarvoor heb je voldoende kennis nodig van het gehele datamodel van ingeschreven personen. Daarnaast moet je van tevoren met de leverancier(s) van de producer(s) afspraken maken over de autorisatie tot deze velden. Dit kost al met al veel tijd.

Met StUF 4.0 is het veel eenvoudiger. Je hoeft als consumer niet meer te vertellen wat je allemaal wil opvragen. Daardoor hoef je niet direct het hele datamodel te kennen. De producer bepaalt eenzijdig waartoe de consumer geautoriseerd is. Daar kun je uiteraard indien nodig nog aanvullende afspraken over maken, maar voor het gros van de situaties heb je aan een standaard set aan basisinformatie voldoende.

Mappen op eigen datamodel

Topicus Overheid heeft in haar consumer een datamodel opgenomen dat (bewust) afwijkt van het RSGB-datamodel. Zo zal het ook vaak zijn in de praktijk, StUF-connectoren worden gebouwd op bestaande applicaties die al een eigen datamodel kennen. Met het Orika-framework (een populair Java bean mapping-framework, zie <http://orika-mapper.github.io/orika-docs/>) is een mapping gemaakt van het gegenereerde naar het eigen datamodel. Door de platte structuur en eenvoudige relaties in het datamodel van StUF 4.0 is dit relatief eenvoudig.

StUF 3.10

Het datamodel van StUF 3.10 kenmerkt zich door het grote aantal objecten, complexe relaties en de aanwezigheid van meta-informatie.

De StUF 3.10 bevragsmodule van Topicus Overheid bevat maar liefst 3294 gegenereerde StUF 3.10 objecten, de StUF 4.0 consumer slechts 41.

Relaties als 'inp.heeftAlsEchtgenootPartner/gerelateerde/inp.bsn/value' zijn korter en daardoor eenvoudiger geworden: 'partner/persoon/geslachtsnaam'.

In StUF 3.10 is het ook vereist voor producers om per element dat leeg is aan te geven waarom dit element leeg is. Dat kan zijn omdat het element simpelweg geen waarde heeft voor de betreffende persoon (het voorvoegsel bijvoorbeeld), of omdat de waarde niet bekend is bij de producer, of omdat de consumer niet geautoriseerd is. Al deze meta-informatie maakt het interpreteren van wat eenvoudige waarden zouden moeten zijn lastig. StUF 4.0 vermijdt dit.